TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 5506 — Introducción al Reconocimiento de Patrones

Proyecto 3
Aprendizaje no supervisado

Fecha de asignación:	24 de Mayo, 2023	Fecha de entrega:	14 de Junio, 2023
Grupos:	3 personas	Profesor:	Jason Leitón Jiménez

1. Objetivo

Implementar un modelo de machine learning relacionado a self supervised, que sea capaz de identificar las clases del set de datos, utilizando alguna arquitectura de autoencoder.

2. Motivación

Los algoritmos de machine learning se presentan cada vez más en los ámbitos cotidianos de la sociedad. La agricultura es uno de los campos que ha incursionado y ha invertido recursos en tecnología relacionada con IA. El machine learning permite analizar grandes volúmenes de datos agrícolas, como información climática, datos del suelo, características de los cultivos, y más. Al aplicar algoritmos de aprendizaje automático a estos datos, se pueden obtener patrones y conocimientos útiles para optimizar las prácticas agrícolas. Esto puede conducir a una mayor eficiencia en el uso de recursos como agua, fertilizantes y pesticidas, lo que a su vez puede reducir los costos de producción.

La IA puede ayudar en la gestión de cultivos al identificar patrones y correlaciones en los datos agrícolas. Por ejemplo, los algoritmos de machine learning pueden detectar patrones en imágenes satelitales o imágenes tomadas por drones para identificar enfermedades de las plantas, deficiencias nutricionales, estrés hídrico y otros problemas. Esto permite una intervención temprana y precisa, lo que lleva a una mejor gestión de los cultivos y una reducción del desperdicio de recursos.

3. Descripción General

Este proyecto consiste en comparar modelos de clasificación de enfermedades de plantas, con un total de 38 categorías. La diferencia es que unos de los clasificadores serán construidos sobre un autoencoder, que a su vez fue entrenado de forma self-supervised con un conjunto de datos **sin labels**. Adicionalmente, se realizará el modelo más robusto usando ruido en la imagen de entrada.

3.1. Set de datos e hipótesis

El set de datos se puede encontrar en https://www.tensorflow.org/datasets/catalog/plant_village. Está conformado por más de 50 000 imágenes divididas entre 38 clases, las cuales hay hojas de plantas con y sin enfermedad. El set de datos permite detectar si hay enfermedad o no por especie de planta. El problema se vuelve en clasificar aquellas hojas que están enfermas.

3.1.1. Hipótesis

1. En ausencia de datos etiquetados, es posible entrenar un autoencoder reconstruyendo datos sin labels, haciendo transfer learning con el subconjunto pequeño que si tiene labels, y obtener mejores resultados que con solo usar el mismo subconjunto pequeño con labels directo para entrenar un clasificador.

2. Al agregar ruido a la entrada del autoencoder, es posible aprender representaciones más robustas que mejoran la clasificación y por ende, la representación latente.

3.2. Experimentos

3.2.1. Experimento 1 asociado a la hipótesis 1

Para el primer experimento se debe tomar el set de datos y simular que una buena parte del mismo no tiene labels. Esto por que en la práctica, es más común tener acceso a datos sin labels que con labels. La parte del set de datos sin labels será usada para aprender a reconstruir los datos (imágenes) con un Autoencoder y aprender una representación latente. Luego, la parte restante con labels, será subdividida en training y testing, para entrenar 2 modelos de clasificación, donde uno de ellos será entrenado desde cero (sin usar nada del autoencoder, ni pesos pre-entrenados), y otro usando el encoder del autoencoder como base. El clasificador que usa el autoencoder como base tendrá a su vez dos variantes: una dejando los pesos aprendidos del encoder congelados durante el entrenamiento del clasificador, y otra permitiendo que cambien.

Se debe crear entonces 2 ejecuciones cambiando los porcentajes lo suficiente para observar la relación de rendimiento de los clasificadores desde cero versus basados en autoencoder, además de comparar el rendimiento si se usa el encoder como feature extractor (congelado) o si puede cambiar (modificando pesos), y sacar sus conclusiones al respecto.

En la tabla 1 se muestra un ejemplo de dos ejecuciones:

Ejecución	Datos sin labels para el encoder	Con labels (training)	Con labels (testing)	Clasificador	Tipo clasificador
1	80 %	10 %	10 %	Clasificador A	Sin autoencoder
				Calsificador B	Pesos del autoencoder congelados
				Calsificador C	Pesos del autoencoder sin congelar
2	50 %	35 %	15 %	Clasificador D	Sin autoencoder
				Calsificador E	Pesos del autoencoder congelados
				Calsificador F	Pesos del autoencoder sin congelar

Cuadro 1: Ejemplo de ejecuciones

Es importante aclarar que se debe usar el mismo corte de training y testing entre clasificadores de la misma corrida para que la comparación sea justa. También se debe usar el split (ambos) de forma estratificada.

3.2.2. Experimento 2 asociado a la hipótesis 2

Se deben comparar dos clasificadores basados en autoencoders, con la diferencia de que uno de los autoencoders fue entrenado agregando algún tipo de ruido a las imágenes de entrada (por ejemplo salt and pepper noise). Este tipo de autoencoder se conoce como Denoising Autoencoder. Se puede reutilizar alguno de los cortes de datos usados en el experimento 1.

Adicionalmente, se debe hacer una visualización de los vectores latentes de las imágenes "sin labels", mostrando los labels (que sí conocemos, solo que no los usamos en el entrenamiento para simular un faltante de labels) con colores o similares (Ver figura 1), para ver si en el espacio latente el modelo aprendió a modelar relativamente bien las clases, es decir, que items de la misma clase estén cerca. Por ejemplo, se esperaría que los vectores latentes del Denoising Autoencoder muestren clusters más claros que los del Autoencoder normal.

Para ello se puede usar alguna técnica de reducción de dimensionalidad para visualización de los vectores, como PCA o t-SNE, un ejemplo de esto se muestra en la figura 1, utilizando tensorboard.

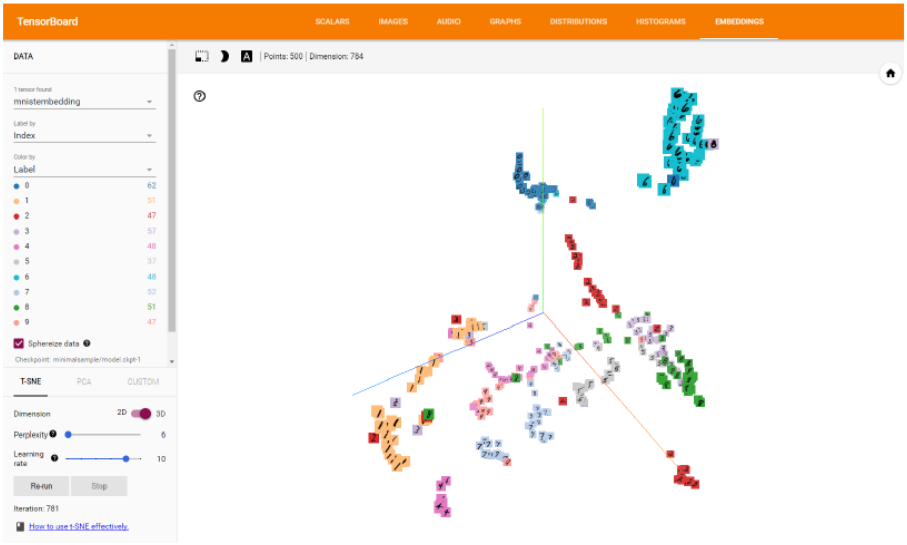

Figura 1: Ejemplo de visualización

3.3. Métricas

Se debe reportar para todos los casos el Accuracy por clase, así cómo las otras métricas derivadas de la matriz de confusión, se debe mostrar el ROC. Es importante destacar que se deben de obtener las métricas del test, sin embargo, las de entrenamiento son necesarias para referirse al overfitting y underfitting.

Las comparaciones entre las métricas deben ser por medio de tablas, en donde sea sencillo la lectura y la evaluación por ejecución.

También se debe de referir sobre las hipótesis y cómo los experimentos las prueban o las contradicen. Si se puede rechazar cada una de ellas o bien si no podemos rechazarlas de acuerdo a la evidencia de los experimentos.

4. Requerimientos técnicos

Las redes neuronales pueden implementarse en las bibliotecas que deseen. Se recomienda el uso de GPUs en Google Colab, subiendo el set de datos en Google Drive. De tener GPU propio, todavía mejor. Otra opción para obtener GPUs es usar el cluster Kabré de CeNAT, o bien en caso de tener la posibilidad, Azure o AWS son opciones viables.

5. Documentación- Estilo IEEE-Trans

- Introducción: Teoría necesaria, breve descripción del proyecto y qué es lo que se espera en el escrito.
- Detalles del diseño del programa desarrollo. En esta sección se espera observar un diagrama de flujo de cada algoritmo, así como la elección de los parámetros a utilizar para cada problema. También los diagramas de arquitecturas de las redes neuronales.
- Resultados y análisis. Se espera ver todas las métricas así como la generalización de las mismas, aquí se debe mencionar los problemas de los set, si hubo overfitting, underfitting, cuál fue el mejor, las razones de los mejores y peores resultados, mencionar comparaciones de resultados con parámetros y cualquier otro análisis que sea relevante
- Conclusiones
- Referencias

6. Entregables

1. Notebook: Un solo notebook(.ipynb) con el código y con los mejores resultados de cada modelo.
2. Paper: PDF con el contenido de la documentación

7. Fecha de entrega y revisión

14 de junio a las 15:00. Revisiones en horario de clase.

8. Evaluación

- Experimento 1 30 %
- Expermiento 2 30 %
- Visualización 10 %
- Paper 30 %